Assertions

Tony Hoare

Redmond. Jan 19, 2001

Why?

- Nostalgia — my research — 1968-
- Importance — 1% of MS code — Now
- Promise — in productivity tools — 2000 -

Purposes

- Test oracle
- Exception generation
- Defect tracking
- Compile-time checking
- Program documentation
- Proofs
- Helping optimisation
- Helping analysis tools
- Preconditions
- Postconditions
- Invariants
- Exceptions

MSO_ASSERT

- Integrated with RAID (and Office Watson)
- Identifies bugs across builds/releases
- Integral to the programming process

In Retail

- VSASSERT assertions are ignored
- VsVerifyThrow … generate exceptions
- VsVerify …user chooses

Explanations

- VSASSERT (assertion, "reason why I think the assertion is true")
- Otherwise it's easy to forget
- Helps both writer and reader.

Compile-time

- #define CASSERT(x) extern dummy[(x)?1:-1]
- Generates report at compile time
- CASSERT (sizeof (x) == sizeof (y))

Documentation

- Protection for system against future changes

- If (a >= b) { … a++ : … };

  ….

  ASSERT (a != b, "a has just been
  incremented to avoid equality") ;

  x = c/(a – b)

Assumptions

- Used only during early test
- X = malloc(1000);
  simplifying_assumption (X != 0 ,
  "replace before check-in by recovery action ")
- Failure indicates test was irrelevant

Invariants

- True of every object …
- …before and after every method call
- bool class_invariant ( )
  {…tests that list is circular…}

PREFIX_ASSUME

- Reduces PREFIX noise

- pointer = find (something);
  PREFIX_ASSUME (pointer != NULL,
  "see the insertion three lines back");
  … pointer ->mumble = blat …

Optimisation

```
switch (condition) {
    case true:  ...  : break;
    case false:  ...  :break;
    default: _assume (false) }
```

- Compiler emits less code

Preconditions

- Obligation is on caller to verify before calling the method

- void insert (node *n) const {

```
entry_assert ( n != NULL );
object_assert (class_invariant( ));
simplifying_assumption (find(n) == 0);
```

... ...

Post-conditions

```
    ...   ...   ...
    object_assert (class_invariant ( ));
    exit_assert ( find(n))
}
```

- obligation on method writer to verify

Crispin Goswell

- Precise Specifications of COM Interfaces and Types.
- Needed for safe re-use of components
- Control dependencies
- Encourage simple interfaces
- Make unit test simple and effective
- Good for user and implementer alike

Summary

- Assertions have many uses
  - … discovered by development teams
  - … and given different notations
  - … and recognised by different tools.

- Ideally, we should use the same assertion for many or all of these purposes.

Life of an assertion

- Discussion point
- Decision
- Task allocation
- Test oracle
- Guarantee by developer
- Compiler hint
- Exception generator
- Long-term documentation

Six Sigma

- Coding Quality Initiative
- Improvement – Write Unit Test First
- Early analysis of potential error
- Seems to help avoidance

Extra variables

- {ASSERT double a0[sizeof(a)] = a ;

  … rearrange a … ;

  exit_assert (a is a permutation of a0)}

- a0 sliced out of retail

Capabilities

- Declare cap_set as an abstract variable holding the set of permitted actions.
- Every action is preceded by an assertion that it is in the cap_set of the current thread.
- Some actions increase or reduce cap_set.
- Tools are needed to reliably insert these assertions and actions.

UTAHlite

- Model-based testing
- Desired behavior abstracted as a graph
- With actions on the edges
- Generates test scripts
- Drives automated test suites

Jason Taylor

ASML

- Abstract State Machine Language
- Powerful abstractions to specify intent
- Analysable, executable as prototype
- Applied to COM components
- Yuri Gurevich…

Promise

- Assertion inference
- Proof of assertions
- Test case generation
- Assertions for concurrency

Michael Ernst

- Dynamic discovery of likely assertions
  by inference from data collected in test
- Gives warning of anomalies
- Estimates test coverage
- Helps when code is changed

Greg Nelson

- Extended Static Checking (of Java)
- Generates verification conditions
- Proves them by decision procedures
- Validates omission of assertions

Rustan Leino

- Annotation inference for modular checkers
- Generates random assertions
- Selects and verifies them by ESC

VAULT

- Enforcing High-Level Protocols in Low-Level Software
- Robert Deline and Manuel Faehndrich
- Graph model tracks capabilities
- Correctness checked at compile time.

Daniel Jackson

- Finding Bugs with a Constraint Solver
- Alloy relational language for specs

Abstraction

- Automatically Validating Temporal Safety Properties of Interfaces.
- Thomas Ball and Sriram K. Rajamani
- Abstracts a program to one with only Boolean variables
- Maybe introduces non-determinism.
- May validate the program or generate a failing test case

Concurrency Safety

- Type-based Race Detection for Java
- Flanagan and Freund

Acknowledgements

Chris Antos, Terry Crowley, Mike Daly, John Douceur, Kirk Glerum, David Greenspoon, Martyn Lovell, Jon Pincus, Hannes Ruescher, Marc Shapiro, Kevin Schofield, David Schwartz, David Stutz